VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Převod CAD formátů na konečně - prvkové sítě a jejich editace

Convertion of CAD Formats into Finite - elements Meshes and their Editation

Jakub Juřina

Bakalářská práce

Vedoucí práce: Ing. Ondřej Meca, Ph.D.

Ostrava, 2021

Abstrakt

Tato bakalářská práce se zabývá převodem CAD formátu na konečně-prvkové sítě a následného připojení na konečně-prvkový řešič ESPRESO. Cílem této práce bylo vybrat a otestovat knihovny pro převod CAD formátu na pro nás vhodnou konečně-prvkovou síť, se kterou by řešič ESPRESO uměl pracovat. Začátek práce je zaměřen na teoretický popis námi zvolených formátu a popisu knihoven vhodných pro řešení. V praktické části byla vytvořena aplikace v jazyce C++ pro testováni knihoven, která byla poté připojena k řešiči ESPRESO.

Klíčová slova

konečně-prvkové sítě; CAD; GMSH; NetGen, Tetgen, ESPRESO

Abstract

This Bachelor thesis deals with converting CAD formats to finite-elements meshes and then connecting to finite-elements solver ESPRESO. The aim of this work was to select and test libraries for converting CAD formats into a suitable finite-elements meshes for us, which the ESPRESO solver could work with. The beginning of the work is focused on the theoretical description of our chosen formats and descriptions of libraries suitable for solutions. In the practical part, a C++ application was created for testing libraries, which was then attached to the ESPRESO solution.

Keywords

finite-elements meshes; CAD; GMSH; NetGen, Tetgen, ESPRESO

Poděkování

Rád bych na tomto místě poděkoval mému vedoucímu bakalářské práce Ing. Ondřeji Mecovi, Ph.D., za jeho rady a připomínky při zpracování této bakalářské práce, protože bez těchto rad a jeho pomoci by tato práce nevznikla.

Obsah

Seznam použitých zkratek a symbolů

SVG	–	Scalable Vector Graphics
VTK	–	Visualization Toolkit
IGES	–	Initial Graphics Exchange Specification
STP/STEP	–	STandard for the Exchange of Product model data
XML	–	Extensible Markup Language
CAD	–	Computer-aided design
ASCII	–	American Standard Code for Information Interchange
OCCT	–	Open Cascade Technology
API	–	Application Programming Interface
STL	–	Standard Triangle Language
GetDP	–	A General Environment for the Treatment of Discrete Problems

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Počítače se v dnešní době staly součástí našeho života a využívá je většina lidí. Počítače se využívají v mnoha oblastech a jednou z těchto oblastí, kde nacházejí široké uplatnění je oblast fyzikálních simulací. Pomocí těchto simulací se dají prověřit vlastnosti nejrůznějších součástek nebo naopak optimalizovat jejich tvar tak, aby daná součástka vyhovovala zadaným kriteriím, aniž bychom jí museli vyrobit. Za tímto účelem už vzniklo mnoho nástrojů, které jsou schopny tyto simulace spouštět. Prvním a nejdůležitějším krokem každé simulace je však vytvoření modelu předmětu pro následnou simulaci. Pro vytvoření tohoto modelu existuje také mnoho nástrojů, které jsou vetšinou postaveny na tvorbě vektorových modelů.

Nástroje pro simulace, ale nejsou schopny pracovat s přímo vygenerovaným vektorovým modelem, a proto se musí nejprve převést do vhodného formátu modelu, se kterým už tyto nástroje umí pracovat. Jedním z možných modelů jsou konečně-prvkové sítě, které slouží jako diskrétní reprezentace původního modelu. Za účelem tohoto převodu vzniklo mnoho nástrojů. Tímto převodem a následným napojením na konečně-prvkový řešič ESPRESO, který je vyvíjen v rámci VŠB na IT4Inovations, se zabývá tato bakalářská práce. Hlavním důvodem vzniku byl fakt, že konečně-prvkový řešič neměl implementovanou možnost načítání CAD formátů.

V první části této bakalářské práce se nachází popis počítačových geometrii, vektorového popisu, který využívají CAD formáty, a konečně-prvkových sítí, jež jsou využity ke konečnému výpočtu na řešiči ESPRESO. Tyto popisy se nachází v následující kapitole 2. V Druhé části bylo třeba projít nástroje, které nám umožňovali generování konečně-prvkových sítí z námi vybraných CAD formátu. Touto problematikou se zabývají kapitoly 3, kde se zabýváme generátory konečně-prvkových sítí a kriterii pro výběr nástrojů, a 4, kde se nachází popis námi vybraných knihoven pro tuto práci. V další části bylo třeba námi vybrané nástroje otestovat, zda jsou vhodné pro následné připojení k řešiči ESPRESO. Tímto se zabývá kapitola 5. V této kapitole je popsané jak se různé knihovny chovají a také jejich další možné nastavování. Poslední část (kapitola 6) se zabývá samotným napojením otestovaných nástrojů na konečně-prvkový řešič.

Kapitola 2

Počítačový popis geometrie

Počítačová geometrie je teoretický základ pro počítačovou grafiku [1], která se zabývá zpracováním grafických informací. Rozdíl mezi geometrií a počítačovou grafikou je například v základních pojmech, kdy geometrie používá pojem bod a počítačová grafika pojem pixel. Pro reprezentaci obrazu v počítačové grafice používáme buď rastrovou nebo vektorovou grafiku. Rastrová grafika je oproti vektorové tvořena sítí pixelů, kdy každý pixel má uloženou vlastní informaci například o barvě nebo jasu. To má za důsledek, že rastrové obrázky mají předem dané rozlišení.

2.1 Vektorový popis

Vektorová grafika [2] je jednou z počítačových grafik, která je definovaná body položenými v prostoru nebo v rovině podle toho, zda používáme 3D nebo 2D, které jsou mezi sebou spojeny čárami a křivkami, a z těch se poté vytváří mnohoúhelníky nebo jiné tvary. Tento popis nám umožňuje popsat jakýkoli tvar například můžeme jednoduše popsat kruh. Ten můžeme vykreslit třeba pomocí jednoho bodu jako středu, druhého bodu ležícího na obvodu kruhu a informace o tom, že vykreslený objekt má být kruh. Stejným způsobem se dá postupovat i u dalších obrazců. Tento typ popisu má oproti rastrové grafice několik výhod:

1. Možnost zmenšovat nebo zvětšovat objekty bez ztráty kvality. Obrázek 2.1 (a) originální vektorový obrázek, (b) zvětšeno 8× jako vektorový obrázek, (c) zvětšeno 8x jako rastrový obrázek.

2. Možnost oddělené práce s objekty.

3. Tím, že je vektorová grafika sestavena ze souřadnic bodu a čarami nebo křivkami mezi nimi, má menší množství informací k vykreslení oproti rastrové grafice, která je definovaná pixelem po pixelu. To znamená, že vektorová grafika mívá menší velikost souboru.

Nevýhody:

1. Při velké složitosti grafického objektu je vetší náročnost na operační paměť a procesor počítače než u rastrové grafiky, protože každý objekt je samostatně definovaný a s větším počtem objektu tedy náročnost vykreslování stoupá

2. Složitější pořízení obrázků oproti rastrové grafiky. Rastrové obrázky se dají pořídit pomocí fotoaparátu nebo například skeneru, ale vektorový obrázek se tímto způsobem získat nedá. Pro vytváření těchto obrázků slouží editory. Například editory firmy Adobe (například Adobe Fireworks [3]), Sketch [4], Inkscape [5], Invertor [6], autoCAD [7].

Vektorová grafika se dnes vyskytuje v několika grafických formátech [8]:

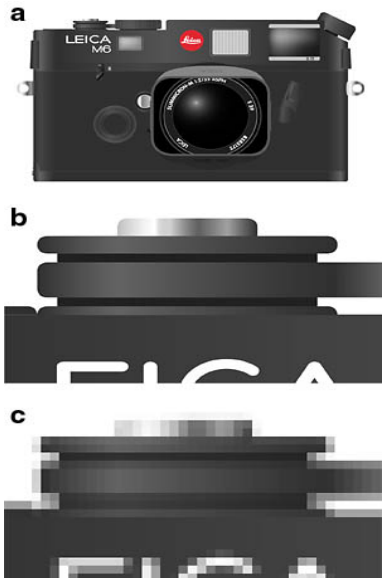
1. SVG [9] - Vektorový formát založený na textových souborech XML. Což umožňuje editaci těchto souborů libovolným textovým editorem. Formát SVG je nejrozšířenější formát 2D vektorové grafiky používaný ve webových prohlížečích. Mezi hlavními výhodami tohoto formátu je to, že ho je možné editovat libovolným textovým editorem, možnou snadnou přenositelnost na různé platformy a celkem jednoduchou čitelnost kódu. Jako nevýhodu můžeme zmínit slabou podporu ve starších verzích prohlížečů.

2. IGES [10] - Vektorový formát pro digitální výměnu dat CAD systémů. Pomocí tohoto formátu se dají přenést jak 2D výkresy tak 3D modely. Jedna z nevýhod tohoto formátu je že neumí přenést kompletní 3D model ale jen jeho povrch, což vede ke ztrátě vnitřní logiky s nimi spojenou ztrátou dat.

3. STEP [11] - Vektorový formát vyvinutý jako nástupce formátu IGES. Tento formát už dokáže přenést kompletní 3D model a tím mizí problémy s nespojitostí námi přenášeného 3D tělesa.

2.2 konečně-prvkové sítě

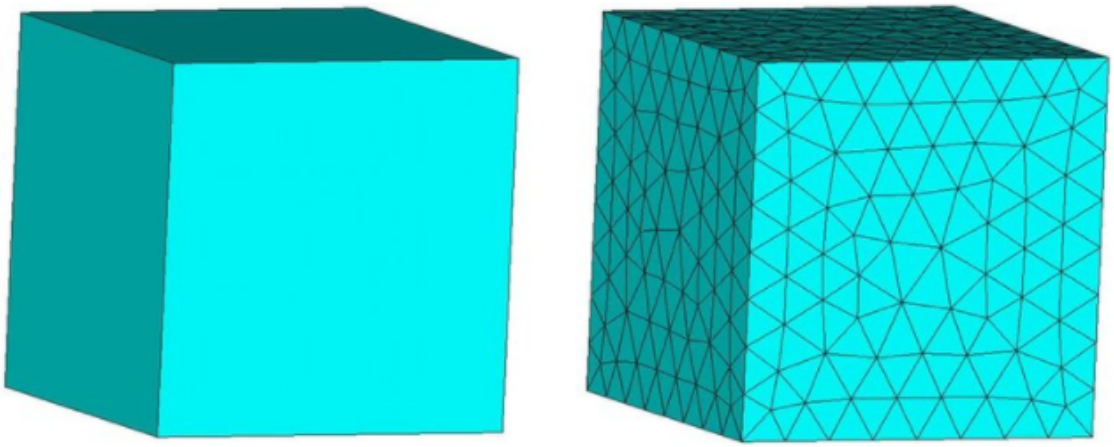
Konečně-prvková sít [12] je diskrétní reprezentace původní geometrie, kterou lze využít například pro vykreslování na obrazovky počítačů nebo pro fyzikální simulaci pomocí parciálních diferenciálních rovnic [13]. Tyto sítě se skládají z jednoduchých buněk jako jsou třeba trojúhelníky, čtyřúhelníky, čtyřstěny, pyramidy, kvádry, atd. Rozdělení geometrie na menší části využíváme z důvodu, že jsme schopni jednodušeji provádět výpočty na zmíněných prvcích než na složitých prostorových vektorově definovaných útvarech. Tyto sítě jsou vytvářeny pomocí různých algoritmů [14], které jsou využívány v programech pro generování těchto sítí. Cílem tohoto generování je aby výsledná síť zachytila co nejpřesnější vstupní geometrii a vytvořila dostatečně jemnou konečně-prvkovou síť pro přesnější výpočty. V níže uvedeném obrázku 2.2 se můžeme podívat jak vlastně tyto konečně-prvkové sítě vypadají. Na levé straně se nachází vygenerovaná kostka a na pravé straně je kostka po použití programu na vytvoření konečně-prvkové sítě.



Obrázek 2.1: Ukázka vektorové grafiky

Jelikož existuje mnoho nástrojů pro práci s konečně-prvkové sítěmi existuje mnoho formátů, jak mohou byt tyto sítě uloženy. Mezi známe formáty patří VTK [15], EnSight [16], ABAQUS [17], OpenFOAM format [18], XDMF [19], atd.

Jednotlivé formáty se liší v mnoha aspektech tak, aby je nastroj s tímto formátem spojeny mohl pohodlně načítat, avšak v základu se každá databáze skládá z definice jednotlivých bodů a elementů. Body jsou popsány souřadnicemi. Elementy jsou popsány jako n-tice bodů v předem definovaném pořadí. Aby bylo možné definovat různé vlastnosti elementu a síly působící na geometrii, tak jsou jednotlivé body i elementy většinou seskupeny do regionu – množin na které se lze dále odkazovat.



Obrázek 2.2: Rozdíl povrchu kostky po použití konečně-prvkové sítě

Kapitola 3

Generátory konečně-prvkových sítí

Za účelem generování konečně-prvkových sítí vznikla cela řada nástrojů, které jsou součástí větších balíku ale i jen jednoúčelové aplikace. Jejich seznam je udržovaný na následujících stránkách [20] nebo [21]. Jako ucelené balíky, které poskytuji komplexní funkcionalitu, můžeme zařadit komerční balíky jako například:

1. ANSA [22] - nástroj pro generování konečně-prvkových sítí široce používaný v automobilovém průmyslu. nástroj byl vyvinut společností BETA CAE Systems.
2. Ansys Workbench [23] - nástroj pro generování konečně-prvkových sítí vyvíjený společností Ansys [24].
3. Solid Edge [25] - nástroj vyvíjený společností Siemens PLM Software [26]. Tento nástroj je schopen vygenerovat a analyzovat konečně-prvkové sítě. Pro toto generování využívá programy Femap a NX Nastran.
4. Commet Solution [27] - Společnost vyvíjející nástroj SimApp, který je založen na webovém GUI.

ale také open-source řešení:

1. Meshlab [28] - nástroj pro zpracování a úpravu 3D sítí vyvíjená Institute of Science and Information Technology s National Research Council v Italii.
2. CGAL [29] - nástroj vyvíjená stejnojmennou společností CGAL Open Source Project. Nástroj se zabývá mnoha tématy jako například generování sítě, rekonstrukce tvarů a další které lze nalézt v dokumentaci [30].
3. Salome Mesh [31] - nástroj, který poskytuje platformu pro práci s modelováním CAD formátů a jejich následné zpracování pro simulaci. Nástroj je vyvíjený společnosti Open Cascade [32], EDF [33] a CEA [34]

Tyto knihovny jsou využívané velkou komunitou uživatelů, což z nich dělá dobře otestované a tedy i robustní nastroje. Pro naše využití v open-source knihovně pouze pro generování sítě však nejsou vhodné, jelikož se většinou jedna o nástroje s vlastním GUI a neposkytují vhodné C API, na které by se dalo napojit, popř. pro generování sítě využívají externí nástroje.

Mnoho knihoven z uvedených seznamu [20, 21] je již zastaralých a nejsou nadále vyvíjené, což znesnadňuje řešení potenciálních problémů. Proto jsme se v této bakalářské práci zaměřili pouze na knihovny, které jsou aktivně vyvíjené a mají aktivní uživatele. Toto kriterium jsme zařadili z důvodu možných problémů, kdy je výhodou, že je knihovna používána větší komunitou lidí, a je tedy menší pravděpodobnost že bychom narazili na nějaký problém a pokud bychom na nějaký problém narazili v dané verzi je více pravděpodobné, že na tuto chybu už někdo před námi narazil a řešení by tedy bylo lehčí. Výhodou také bylo, pokud existovala dokumentace, která nám pomohla se s knihovnou blíže seznámit.

Podle těchto kriterii jsme nakonec vybrali 3 knihovny, které jsme otestovali (gmsh [35], netgen [36], tetgen [37]). Tyto knihovny jsou popsány blíže v následujících kapitolách.

Kapitola 4

Vybrané knihovny

V této kapitole si blíže popíšeme knihovny, které jsme vybrali z knihoven uvedených v předchozí kapitole 3. Z těchto knihoven bylo třeba vybrat ty, které by byly nejlepší pro naše účely a práce s těmito knihovnami by nebyla náročná. Při výběru jsme začali pracovat prvně s knihovnou GMSH, kterou dále popisujeme v kapitole 4.1. Po Knihovně GMSH si přiblížíme knihovnu Netgen (Kapitola 4.2), se kterou jsme začali pracovat po dotestování knihovny GMSH. Jako poslední jsme zvolili knihovnu Tetgen, která je popsána v kapitole 4.3. Jako poslední částí této kapitoly je popis knihovny OpenCASCADE (kapitola 4.4) potřebné pro práci se STP/STEP a IGES soubory. Každá z námi vybraných knihoven obsahuje vlastní textovou dokumentaci GMSH [38], Netgen [39] a Tetgen [40].

4.1 GMSH

Námi první vybranou knihovnou je knihovna GMSH, která byla vyvinuta Christophe Geuzaine a Jean-François Remacle. Vývoj této knihovny započal v roce 1997 a v této chvíli se nachází ve verzi 4.8.4 a je stále vyvíjená. Knihovna GMSH je také součástí projektu GetDP [41].

Gmsh je knihovna pro generovaní konečně-prvkových sítí nabízející grafické uživatelské rozhraní a také programovací API pro propojení s vlastním kódem. Toto API je napsáno různých jazycích například Python, Julia, C a C++, které je pro naše účely nejdůležitější. Knihovna nám nabízí 2 možnosti instalace. Jako první možnost lze požít k instalaci stažení zdrojového kódu [42]. Jako druhou možností je instalaci balíčku operačního systému linux pomocí příkazu: *sudo apt-get install libgmsh-dev*.

Tato knihovna obsahuje 4 moduly. Jako první modul je geometrie. Ten slouží k vytvoření modelu entit pomocí ohraničení objemu povrchem, ten je ohraničen křivkami a ty dvěma koncovými body. Jako druhý modul obsahuje generátor sítě konečných prvků. Tento modul generuje sít z načtené geometrie pomoci jednoduchých geometrických útvarů jako jsou například čáry, trojúhelníky, čtyřstěny, hranoly nebo třeba pyramidy. K tomuto generování má implementované algoritmy, které jsou popsány níže. Jako další modul obsahuje rozhraní externího řešitele, který je řešen pomocí

implementování ONELAB. A jako poslední modul je vizualizační nastroj k následnému znázornění a zpracování dat.
Výhody:

1. Velký počet možných parametrů pro generování konečně-prvkových sítí. Popis všech parametru se nachází v dokumentaci [38] v sekci B.3 Mesh options list.

2. Díky přiloženým testovacím souborům, napsaných ve výše uvedených jazycích, na testování této knihovny a přiložené dokumentaci, byla implementace k vlastnímu kódu lehčí.

3. V případě potřeby knihovna obsahuje knihovna metody na dodatečnou úpravu vygenerované sítě.

4. Možnost načtení vstupních souborů formátu STP/STEP. Pro práci s těmito soubory je třeba knihovna Open Cascade Technology (více kapitola 4.4).

Nevýhody:

1. Složitější přístup k vygenerovaným datům, potřebných k následujícímu uložení popsaný v kapitole. 5.1

4.2 Netgen

Netgen je námi druhá vybraná knihovna, která je součástí projektu NGSolve. Na vyvíjení tohoto projektu se podílí univerzity: Vienna University of Technology, University of Göttingen, Portland State University, RWTH Aachen University, Johannes Kepler University. Od března roku 2017 se začaly vydávat měsíční aktualizace pro Netge a NGSolve za účelem stability. V současné době se systém nachází v šesté verzi.

Tato knihovna slouží pro generování konečně-prvkových sítí pro 2D a 3D. Tato knihovna lze požít pomocí vlastního grafického uživatelského rozhraní nebo jej lze připojit k vlastnímu kódu pomocí C++ knihovny. Stejně jak u předchozí knihovny má i tato 2 možnosti instalace. První možnost je pomocí zdrojového kódu. Tento způsob je popsán v dokumentaci [43], a druhou možností je instalace balíčku operačního systému linux pomocí příkazu: *sudo apt-get install ngsolve*. Knihovna generuje trojúhelníkové nebo čtyřúhelníkové sítě pro 2D výstupy a čtyřboké sítě pro 3D výstupy.
Výhody:

1. Lehčí postup pro generování konečně-prvkové sítě, který je popsaný v dokumentaci pro vstupní soubor formátu STL. Tento postup se minimálně liší od generování konečně-prvkové sítě pro vstupní soubory formátu STP/STEP.

2. V případě potřeby knihovna obsahuje metody na dodatecnou úpravu vygenerované sítě.

3. Lehké získání vygenerovaných dat proti knihovně GMSH.

4. Možnost načtení vstupních souborů formátu STP/STEP. Pro práci s těmito soubory je třeba knihovna Open Cascade Technology (více kapitola 4.4).

Nevýhody:

1. Menší počet parametrů pro generování konečně-prvkových sítí oproti knihovně GMSH.

4.3 Tetgen

Tetgen byla poslední vybraná knihovna, která byla vyvinuta jako součást výzkumného projektu podporovaného Weierstrassovým institutem pro aplikovanou analýzu a stochastiku (WIAS) zaměstnancem Hang Si. Knihovna slouží pro generování pouze 3D konečně-prvkových sítí. Na generování sítě požívá pouze elementy typu čtyřstěnu. Hlavní předností této knihovny je v generování konečně-prvkových sítí pomocí příkazové řádky, kde je možnost nastavení mnoha parametrů (více dokumentace [40]). Pro naše účely je součástí této knihovny také možnost připojit tuto C++ knihovnu k našemu vlastnímu kódu. Součástí Tetgenu je také vlastní zobrazovací nástroj pojmenovaný TetView.

4.4 Open CASCADE Technology

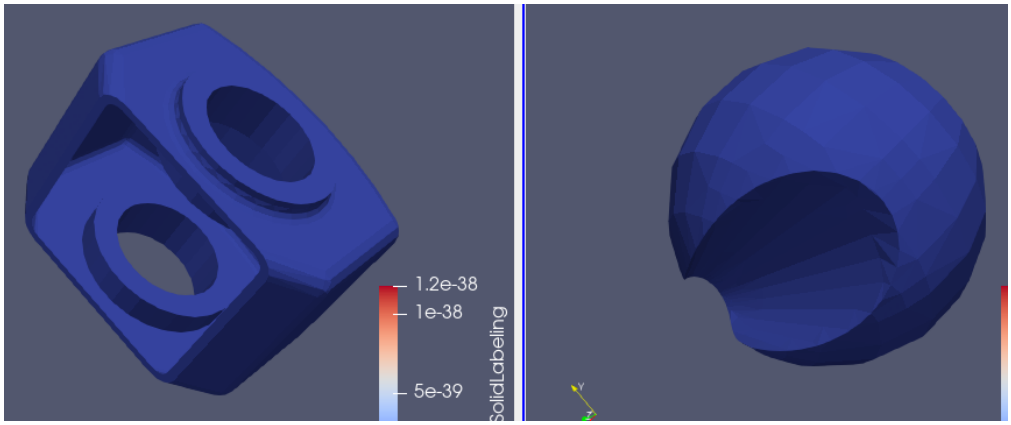
Open CASCADE Technology (OCCT) [44] je platforma pro vývoj softwaru 3D modelování povrchu a těles, výměnu dat CAD systému a vizualizaci. OCCT je vyvíjená od roku 1999 vývojovou společností Open Cascade S.A.S [32]. Pro naše potřeby je OCCT k dispozici ve formě C++ knihovny, kterou lze volně stáhnou a nainstalovat podle internetové dokumentace [45]. Na OCCT je založených několik CAD programu například námi používaný GMSH, dále také FreeCAD, KiCad a programy vyvinuté společností Open Cascade jako třeba CAD Builder či CADRays.

Kapitola 5

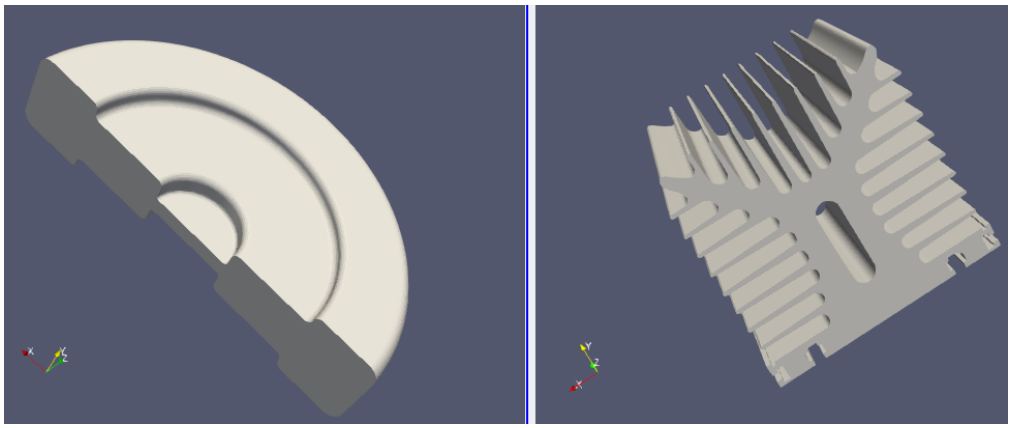
Testováni knihoven

V této kapitole se budeme věnovat testování výše zmíněných knihoven. Pro účely tohoto testovaní jsme si vytvořili jednoduchou aplikaci, která uměla načíst vstupní geometrie z námi zvoleného formátu STL, STP/STEP a IGES, které byly poté poslány do vybrané knihovny, která vytvořila konečně-prvkovou sít. Námi vygenerovanou sít jsme poté ukládali do souboru formátu VTK [15], který je popsaný níže v kapitole 5.0.1. Ukládaní do VTK bylo zvoleno z důvodu nahlédnutí na výsledek a pozdějšího lehčího napojení na konečně-prvkový řešič ESPRESO, který využívá podobný formát vstupních dat.

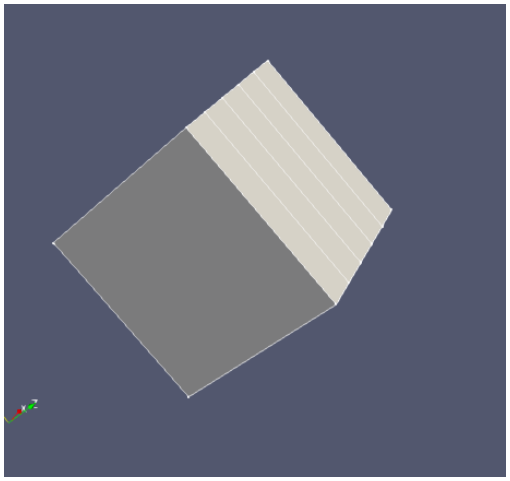
Pro testování knihoven jsme v prvním případě zvolili pro knihovnu Netgen základní nastavení a pro knihovnu GMSH jsme zvolili nastavení z přiloženého testovacího souboru t13, který se nacházel u zdrojového kódu této knihovny, kde byla hodnota parametru MathEval Field na 4, udávající velikost prvků. Tento parametr je popsaný níž. Toto testování probíhalo na souborech zobrazených na obrázku 5.1. Když se tedy podíváme do níže uvedené tabulky 5.1, tak si můžeme na první pohled všimnout, že knihovna Netgen má přímo úměrné chování vzhledem ke složitosti geometrie. To znamená, že čím je geometrie složitější, tím se zvedá počet vygenerovaných bodů a elementů. Na rozdíl od toho se můžeme podívat na tabulku knihovny GMSH, kde jde vidět, že při námi zvoleném nastavení není knihovna schopna zachytit složitost všech nastavených geometrii, protože knihovna GMSH je závislá na absolutní velikosti zpracovaného modelu. Dále si taky můžeme všimnou, že časové průběhy knihovny Netgen jsou rychlejší než od knihovny GMSH. Mohli bychom říct, že tím, že GMSH nebylo schopno zachytit tuto složitost, tak nad touto geometrií pracoval déle, ale při nastavení parametru na hodnotu aby zachytila tuto geometrii se doba generování ještě více zvedla (tuto skutečnost lze vidět ve 3. tabulce). Toto chování můžeme vidět na obrázku 5.2.



(a) Soubory t13.stl a object.stl



(b) Soubory wheel.stl a cooler.stl



(c) Soubor Cube.stp

Obrázek 5.1: Testované soubory

Název souboru	Čas	Počet bodů	Počet elementů
cooler.stl	1009,8 sec.	1569	8680
wheel.stl	245,6 sec.	130	599
object.stl	1,1 sec.	147	689
t13.stl	15,8 sec.	3558	18257
Cube.stp	1,2 sec.	373	1918

(a) Tabulka pro GMSH při nastavení pomoci souboru t13

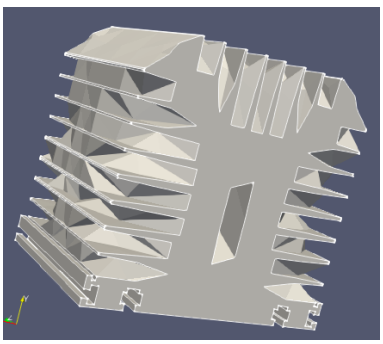
Název souboru	Čas	Počet bodů	Počet elementů
cooler.stl	213,2 sec.	17761	70314
wheel.stl	71,7 sec.	5020	22696
t13.stl	6,5 sec.	1589	4609
Cube.stp	0,15 sec.	21	28

(b) Tabulka pro Netgen v základním nastavení

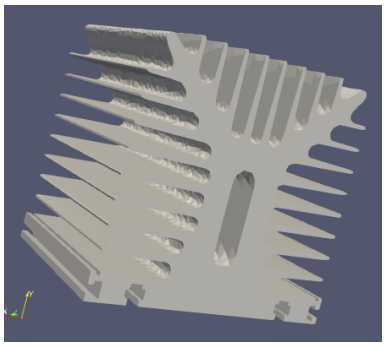
Název souboru	Čas	Počet bodů	Počet elementů	hodnota parametru MathEval Field
cooler.stl	4545 sec.	20939	112548	0,004
wheel.stl	3665 sec.	7355	41842	0,02
object.stl	1 sec.	123	347	
t13.stl	15,8 sec.	3558	18257	4
Cube.stp	1,2 sec.	373	1918	4

(c) Tabulka pro GMSH při změně parametru MathEval Field

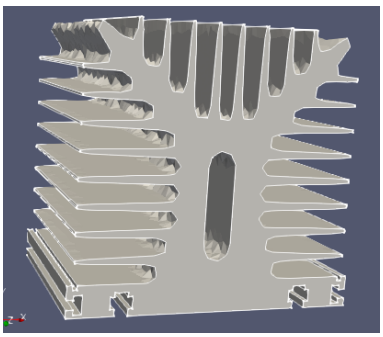
Tabulka 5.1: Tabulky testování knihoven GMSH a Netgen



(a) GMSH cooler s hodnotou MathEval Field = 4



(b) Netgen coller



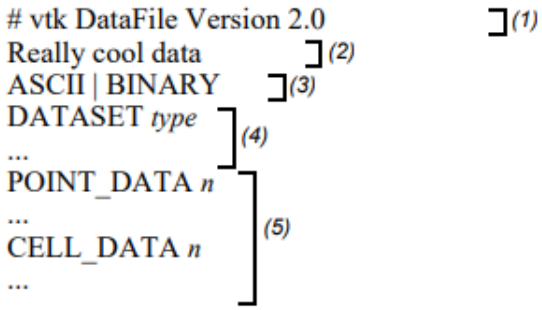
(c) GMSH coller s hodnotou MathEval Field = 0.007

Obrázek 5.2: konečně-prvkové sítě souboru coller.stl

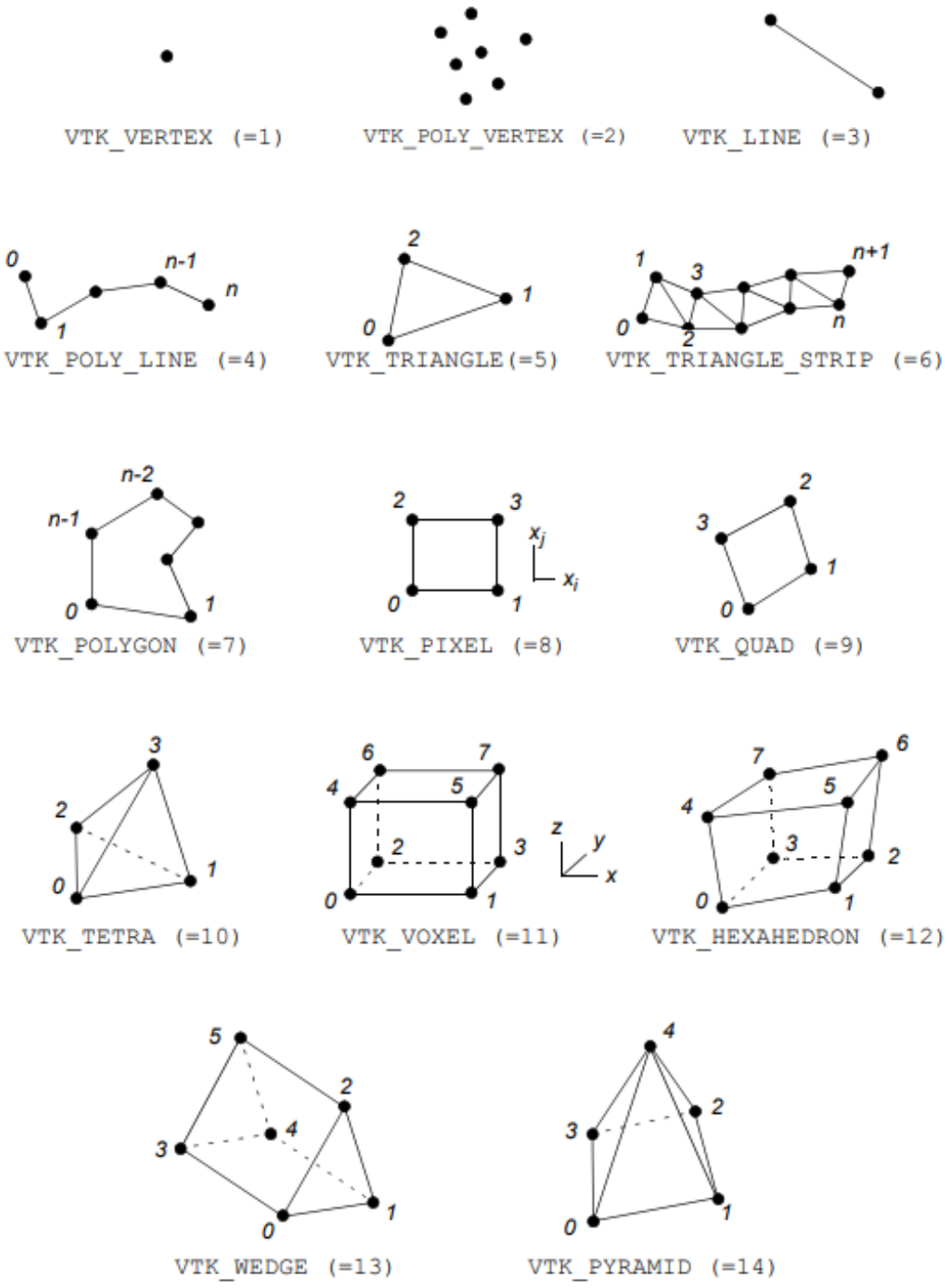
5.0.1 VTK

Pro ukládání výsledných dat do tohoto formátu jsme se rozhodli z důvodu jednoduchosti zápisu tohoto formátu a také kvůli podobnosti s formátem, který je používaný knihovnou ESPRESO (více v kapitole 6). Pro popis formátu VTK použijeme obrázek 5.3.

1. Hlavička souboru určující verzi.
2. Krátký popis souboru v maximální délce 256 znaků.
3. Datový typ zápisu dat. Pro naše účely jsme zvolili ASCII.
4. Část pro popis typu datové sady. VTK nabízí pro ukládání dat 6 možností s tím, že následující část je u každé z možností rozdílná. V našem bylo nutné použít volbu UNSTRUCTURED_GRID (nestrukturovaná mříž), protože jsme generovali obecné (nestrukturované) sítě.
5. Poslední část je u tohoto typu rozdělena do tří částí. První část obsahuje na první řádku slovo POINTS za kterým je celkový počet bodů a pote datový typ bodu (například: POINTS 373 double). Na dalších rádcích je výpis souřadnic bodu x, y, z. Po zapsaní všech bodu následuje prázdný řádek a poté zápis elementů, který je rozdělen na dvě části – nejprve body, z kterých se elementy skládají, poté typy elementů. První řádek z této části obsahuje klíčové slovo CELLS, které je následováno celkovým počtem elementů a za tímto číslem následuje počet zapsaných čísel (například: CELLS 1918 8798). Jednotlivé elementy se skládají z počtu uzlů elementu následovaného výčtem těchto uzlů. (například: přímka složená ze dvou bodu se zapíše: 2 0 1, kde 2 je počet uzlů přímky a 0 1 popisují pozice bodů v sekci POINTS, které jsou na vytvoření použity). Poslední část zápisu obsahuje popis typu elementů. Popis začíná CELL_TYPES, za kterým je celkový počet elementů (například: CELL_TYPES 1918). Každý typ má jiné číselné kódové označení (například. 1= bod, 3= přímka, 5= trojúhelník, 14= jehlan, více v obrázku 5.4.).



Obrázek 5.3: Popis souboru formátu VTK



Obrázek 5.4: Ukázka typu elementů s očíslováním pro soubor VTK

5.1 GMSH

Pro využití této knihovny je potřeba hlavičkový soubor *gmsh.h* a knihovna *libgmsh*, které získáme výše uvedenou instalací. Hlavičkový soubor obsahuje řadu funkcí které by se daly rozdělit do několika kategorií: funkce na vytvoření vlastní geometrie v programu, načtení vstupních souborů, editaci, generování konečně-prvkové sítě a načtení výsledků. V našem případě jsem požívali pro nečtení souboru s příponou STL, STP/STEP a IGES funkce:

1. gmsh::merge(string file); - Funkce pro načtení souboru formátu STL.
2. gmsh::model::occ::importShapes(string file, vector<pair<int, int> > v); - Funkce pro načtení souboru formátu STP/STEP nebo IGES. Hodnota druhého parametru je prázdná a slouží pro možné následující práce s objektem, které nevyužíváme.

Po načtení vstupního souboru se provede nastavení geometrie pomocí funkce gmsh::model::mesh::classifySurfaces(angle * M_PI / 180., includeBoundary, forceParametrizablePatches);, kde první parametr slouží k rozdělení povrchu geometrie podle uhlů hran plochy, který tímto parametrem nastavíme. Druhý parametr vytvoří křivky na na hranicích otevřeného prostoru a třetí parametr slouží k vytvoření hran a povrchů, které lze znovu parametrizovat. Poté následuje nastavování parametrů, které budou popsány níže, a nakonec se vygeneruje konečně-prvková síť. Z této vygenerované sítě, byly potřeba získat potřebné data, která byla pro další práci složitější k získání. V níže uvedeném kódu se můžeme podívat, že před uložením bylo třeba prvně projít všechny entity modelu (entity - jedná se o regiony, do kterých se geometrie rozdělí, například podle toho na jakou hodnotu nastavíme velikost úhlu ve výše zmíněné funkci), kde jsme jednotlivá data ukládali do dočasných proměnných, které jsme poté použili pro zápis do souboru formátu VTK. Bližší popis je uveden pomocí komentářů v kódu.

```
// Část kódu pro získání dat k uložení konečně-prvkové sítě do souboru VTK
void writeGMSH(string out){
   int points=0,element=0,element1=0; // Dočasné proměnné
   vector<double> nodecoord; // Dočasná proměnná
   vector<pair<int, int>> form; // Dočasná proměnná
   vector<size_t> help; // Dočasná proměnná
   vector<pair<int, int> > entities;
   gmsh::model::getEntities(entities); // Funkce pro získání všech entit geometrie
   for(unsigned int i = 0; i < entities.size(); i++) {
      vector<size_t> nodeTags;
       vector<double> nodeCoords, nodeParams;
       int dim = entities[i].first, tag = entities[i].second;
       gmsh::model::mesh::getNodes(nodeTags, nodeCoords, nodeParams, dim, tag); //
            Funkce pro získání informací o bodech dané entity. například souřadnice
```

```
        points=points+nodeTags.size(); // Součet pro celkový počet bodů
        for (int k=0;k<nodeCoords.size();k++){
          nodecoord.push_back(nodeCoords.at(k)); // Zápis souřadnic do dočasné promě
              nné
        }
        vector<int> elemTypes;
        vector<vector<size_t> > elemTags, elemNodeTags;
        gmsh::model::mesh::getElements(elemTypes, elemTags, elemNodeTags, dim, tag)
            ; // Funkce pro získání všech elementu určité entity
        int numElem = 0;
        for(unsigned int j = 0; j < elemNodeTags.size(); j++){
          for(unsigned int k = 0; k < elemNodeTags[j].size(); k++){
            help.push_back(elemNodeTags[j][k]-1); // Zápis bodů že kterých je
                element tvořen do dočasné proměnné
            element1++; // Počítadlo těchto bodů
          }
        }
        for(unsigned int j = 0; j < elemTags.size(); j++){
             numElem += elemTags[j].size(); // Počet elementů jedné entity
        }
        element=element+numElem; // Součet elementu pro všechny entity
        string type;
        gmsh::model::getType(dim, tag, type);
        vector<int> partitions;
        gmsh::model::getPartitions(dim, tag, partitions);
        for(unsigned int j = 0; j < elemTypes.size(); j++) {
          string name;
            int d, order, numv, numpv;
            vector<double> param;
            gmsh::model::mesh::getElementProperties(elemTypes[j], name, d, order,
                numv, param); // Funkce pro získání typu elementu
            form.push_back(make_pair(numElem,elemTypes[j])); // Zápis počtu typu
                elementu a počtu kolik techto elemetů je
      }
    }
}
```

Listing 5.1: Funkce na získání dat z knihovny GMSH

V nastavení z přiloženého testovacího souboru t13 má tato knihovna delší časové průběhy a u některých načtených geometrií si nedrží svou kvalitu oproti knihovně Netgen.

5.1.1 Parametry

Knihovna GMSH nabízí velké množství nastavovacích parametrů k vygenerování sítě. Parametrů bylo velké množství, proto jsme se zaměřili pouze na ty, které byly pro naše účely generování sítě pro konečně-prvkový řešič nejdůležitější, protože měly na výslednou vygenerovanou konečně-prvkovou síť největší vliv. Většina parametrů se tedy zaměřuje na změnu hrubostí generování sítě, protože jak jsme mohli vidět v úvodu této kapitoly, tak nevhodným nastavením parametrů nebyla vygenerovaná síť schopna zachytit danou geometrii. Většina popsaných parametrů se volá funkcí setNumber (gmsh::option::setNumber("Mesh...", číslo);). Jako další možnost pro úpravu vygenerované konečně-prvkové sítě nám GMSH nabízí funkci refine, která se volá pomocí příkazu gmsh::model::mesh::refine();. Tato funkce už jen upravuje vygenerovanou sít a není tedy schopna zaznamenat detaily ztracené špatně nastavenou jemností. Většina parametrů nemá na čas potřebný pro vygenerování konečně-prvkové sítě výrazný vliv. Změna přichází při nastavování parametrů Algorithm3D. Při nastavení na hodnotu 10 (viz 2), se čas generování zrychlí. Čas se také mění s hodnotami parametrů LengthMax a LengthMin. Pokud docílíme nastavením těchto dvou parametrů, že vygenerovaná síť bude jemnější, než při nastavení podle příkladu t13 t, tak se začne čas generování prodlužovat.

1. MathEval Field - Tento parametr se nastavuje pomocí příkazu:
   gmsh::model::mesh::field::setString(f, "F", "4");. Tímto parametrem, který je jako 3. parametr volané funkce, se nastavuje jak jemnost sítě, tak zachycení složitosti geometrie, jak jsme mohli vidět na obrázku 5.2.

2. Algorithm3D - Parametr určující algoritmus pro vygenerování konečně-prvkové sítě. Základní nastavení tohoto parametru je 1 => Delaunay algoritmu. Dále lze požít Frontal algoritmus při nastavení na hodnotu 4, který postupuje při generovaní konečně-prvkové sítě od krajů ke středu a využívá jej knihovna Netgen. Nastavením na hodnotu 7 se požije algoritmus knihovny MMG3D, který má delší časový průběh. Při nastavení parametru na hodnotu 10 se zavolá algoritmus HXT, který vytvoří konečně-prvkovou sít na principu paralelního Delaunay algoritmu, díky kterému generuje výsledek rychleji. Při nastavení parametru na hodnotu 3 a 7 aplikace neprovedla vygenerování sítě.

3. LenghtFromPoints - Parametr, který vypočítá velikost prvků vygenerované konečně-prvkové sítě z hodnot uvedených v bodech geometrie. Parametr je v základu nastaven na hodnotu 1

(Obrázek 5.5a). Pokud změníme nastavení na 0 (Obrázek 5.5b) výsledné generované sítě se razantně zhorší povrchová sít a tím pádem nezachycuje původní geometrii přesně.

4. LenghtFormCurvature - Parametr k výpočtu velikosti prvků vygenerované konečně-prvkové sítě ze zakřivení. Při nastavení parametru na kladnou hodnotu je vygenerovaná sít jemněji generovaná a tím pádem i hustší.

5. LengthExtendFromBoundary - Parametr pro výpočet velikosti elementu. Pro tento výpočet velikosti používá délku hrany. Při nastavení na hodnotu 1 nejdelší a při nastavení na hodnotu 2 nejkratší hrany elementu na povrchu. Výsledek tohoto parametru se viditelně neprojeví na povrchu vygenerované sítě, ale vnitřní struktura se při nastavení na hodnotu 2 pozmění a zvetší se počet generovaných nodu a elementu.

6. LengthMax - Parametr pro určení maximální velikosti elementu. Tento parametr je nejlepší kombinovat s parametrem LengthMin. Bližší popis u dalšího parametru LengthMin.

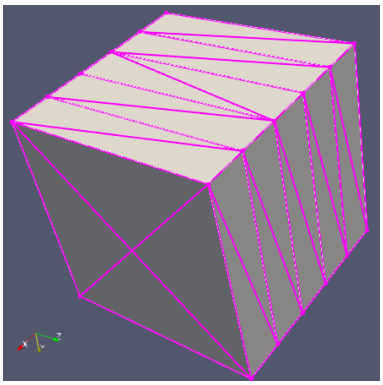
7. LengthMin - Parametr pro určení minimální velikosti elementu. Čím víc se hodnota tohoto parametru blíží k nule, tím je velikost elementu menší. Generování konečně-prvkové sítě se dá tedy ovlivnit jak k jemnějšímu, tak k hrubšímu výsledku . Při spojení s parametrem LengthMax se dá nejvíce ovlivnit velikost vygenerované sítě. Mění se struktura jak vnější, tak vnitřní struktury sítě. Na níže uvedených obrázcích se můžeme podívat na změnu, kterou tento parametr vytvořil. Obrázek 5.5e je v základním nastavení a obrázek 5.5h je síť vygenerovaná nastavením parametrů LengthMax a LengthMin na hodnotu 2. Při nastavení parametru na hodnotu, která začíná generovat síť jemnější, se začíná prodlužovat čas generování.

8. Optimize - Parametr na určení počtu optimalizací vygenerované konečně-prvkové sítě. Vetším počtem optimalizací se zlepší kvalita čtyřstěnných elementů a kvalita generované sítě. Zlepšením kvality elementu se stane, že zaniknou některé elementy.

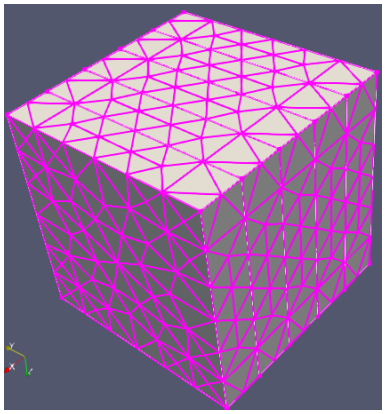
9. OptimizeNetgen - Podobný parametr jak Optimize akorát volá na optimalizaci fukci od Netgenu k lepší kvalitě elementů.

10. SubdivisionAlgorithm - Parametr, který rozdělí elementy vygenerované konečně-prvkové sítě na menší. Nastavení hodnoty 1 se upravují 2D sítě a generují se čtyřúhelníkové elementy místo trojúhelníkových elementů. Tato hodnota je tedy pro naše účely nepotřebná. Pokud nastavíme hodnotu na 2, která slouží pro 3D, generují se elementy typu šestistěnů.

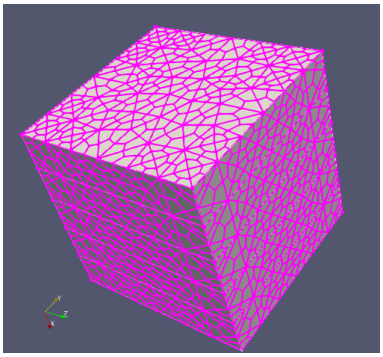
11. Refine - Funkce na rozdělení všech elementů. Tím se vygenerovaná konečně-prvková sít stává o dost jemnější, ale větší přesnost má dopad na dobu generování sítě, která je s rostoucí složitostí delší. Rozdíl s použitím refine můžeme vidět na obrazcích 5.5d a 5.5g, kde druhá sít je vygenerována s použitím popisovaného parametru. Ve většině případů tady tuto funkci využít nemusíme, protože výsledná geometrie se dá lehce ovlivnit výše popsanými parametry.



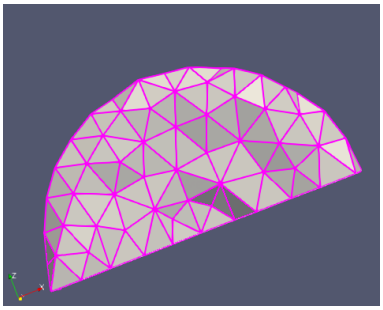
(a) konečně-prvková sít souboru Cube



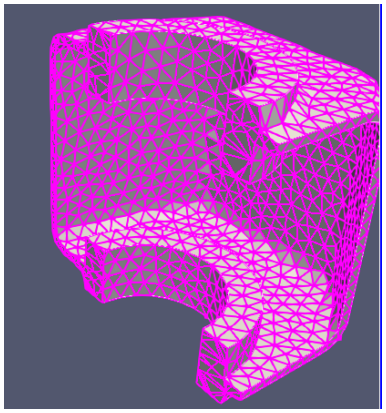
(b) konečně-prvková sít souboru Cube s parametrem LenghtFromPoints



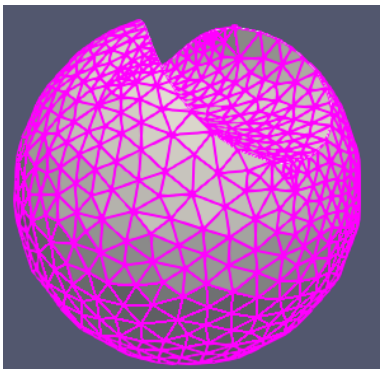
(c) konečně-prvková sít souboru Cube s parametrem SubdivisionAlgorithm



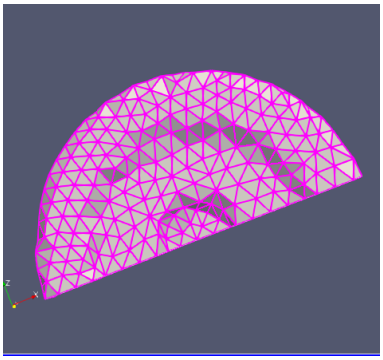
(d) konečně-prvková sít souboru wheel



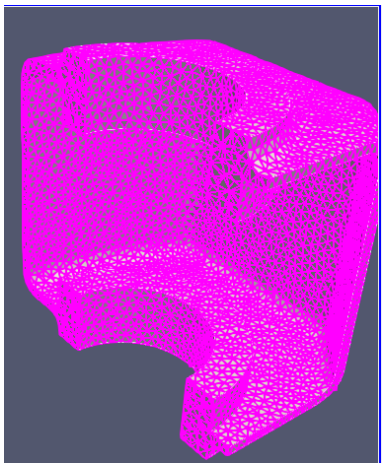
(e) konečně-prvková sít souboru t13



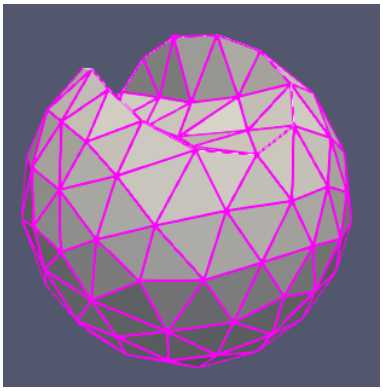
(f) Konečně-prvková sít souboru object s parametrem LenghtFormCurvature



(g) Konečně-prvková sít souboru wheel s parametrem Refine



(h) Konečně-prvková sít souboru t13 s parametrem LengthMax a LengthMin



(i) Konečně-prvková sít souboru object

Obrázek 5.5: Testované parametry knihovny GMSH

5.2 Netgen

Stejně jako u předchozí knihovny je potřeba pro práci přilinkovat hlavičkový soubor *nglib.h* a také knihovnu *libnglib*, které získáme instalací této knihovny, jež je popsána v kapitole 4.2. Hlavičkový soubor také obsahuje funkce na načítání vstupních souboru, editaci, generování konečně-prvkových sítí a získání vygenerovaných výsledků. Knihovna má oproti předchozí knihovně GMSH jednodušší použití. Pro načtení vstupních souboru jsem požili tyto 3 funkce:

1. Ng_STL_LoadGeometry(char*) - Funkce pro načtení vstupního souboru formátu STL.
2. Ng_OCC_Load_STEP(char*); - Funkce pro načtení vstupního souboru formátu STP/STEP.
3. Ng_OCC_Load_IGES(char*); - Funkce pro načtení vstupního souboru formátu IGES.

Lze si všimnout, že na rozdíl od předcházející knihovny, tato požaduje na vstupu funkce řetězec datového typu char, proto jsme museli název převést z datového typu string na námi potřebný. Po načtení vstupního souboru se zavolají funkce pro generování konečně-prvkové sítě, do kterých se také posílají kromě načtené geometrie i parametry pro práci s geometrii. Jak bylo dříve řečeno, je tato knihovna jednodušší k použití a to hlavně při získání potřebných dat na zápis do souboru formátu VTK. Na tento zápis se můžeme podívat v níže uvedeném kódu.

```
// Část kódu pro zápis konečně-prvkové sítě z knihovny Netgen
int main(int argc, char **argv)
{
    Ng_Mesh *mesh = Ng_NewMesh();
    ofstream sfile;
   sfile.open(save);
    int points = Ng_GetNP(mesh), elements = Ng_GetNE(mesh); // Funkce pro získání
        celkového počtu bodů a elementů

    sfile << "# vtk DataFile Version 2.0" << endl; // Hlavička souboru
   sfile <<"NGsolve, Created by NGsolve" << endl;
   sfile <<"ASCII" << endl;
   sfile <<"DATASET UNSTRUCTURED_GRID" << endl;
   sfile <<"POINTS " << points << " double" << endl; // Zápis celkového počtu bodu
        a datového typu

   double point[3];
   for (int p = 1; p <= points; p++) {
         Ng_GetPoint(mesh, p, point); // Funkce na vygenerování souřadnic určitého
            bodu
```

```
      sfile << point[0] << " " << point[1] << " " << point[2] << endl; // Zápis
          souřadnic do souboru
   }
sfile << endl;
int element[NG_VOLUME_ELEMENT_MAXPOINTS], enodes = 0;
for (int e = 1; e <= elements; e++) { // Cyklus pro zjištění celkového počtu
    bodů, ze kterých jsou elementy vytvořeny
   switch (Ng_GetVolumeElement (mesh, e, element)) {
      case NG_TET: enodes += 4; break;
      case NG_PYRAMID: enodes += 5; break;
      case NG_PRISM: enodes += 6; break;
      case NG_TET10: enodes += 10; break;
      }
}
sfile << "CELLS " << elements << " " << elements + enodes << endl; // Zápis
    celkového počtu elementů a počtu bodů, ze kterých jsou elementy vytvořeny
for (int e = 1; e <= elements; e++) {
   switch (Ng_GetVolumeElement (mesh, e, element)) {
      case NG_TET: enodes = 4; break;
      case NG_PYRAMID: enodes = 5; break;
      case NG_PRISM: enodes = 6; break;
      case NG_TET10: enodes = 10; break;
   }
   sfile << enodes; // Zápis z kolika bodů je element vytvořen
   for (int en = 0; en < enodes; ++en) {
      sfile << " " << element[en] - 1;
   }
   sfile << endl;
}
sfile << endl;
sfile << "CELL_TYPES " << elements << endl; // Zápis celkového počtu typů
    elementů a následuje zápis daných typů.
for (int e = 1; e <= elements; e++) {
   switch (Ng_GetVolumeElement (mesh, e, element)) {
      case NG_TET: sfile << "10" << endl; break;
      case NG_PYRAMID: sfile << "14" << endl;; break;
      case NG_PRISM: sfile << "13" << endl; break;
      case NG_TET10: sfile << "24" << endl; break;
```

```
    }
  }
}
```

Listing 5.2: Zápis konečně-prvkové sítě Netgen

Hlavní rozdíl oproti předchozí knihovně je v tom, že není třeba prvně procházet celou síť a ukládat potřebné data do dočasných proměnných, ale lze přímo při čtení vygenerovaných dat zapisovat do souboru, kromě druhé části souboru VTK, kde je prvně potřeba zjistit, kolik bude zapsaných v této sekci čísel, protože každý element se skládá z jiného počtu bodů. Knihovna má v základním nastavení rychlé časové průběhy, výsledné vygenerované sítě nejsou v některých případech příliš jemné, ale drží si svou kvalitu.

5.2.1 Parametry

Pro nastavování parametru má knihovna vlastní strukturu Ng_Meshing_Parameters, díky které je toto nastavování jednodušší než u knihovny GMSH, protože jsou parametry popsány v hlavičkovém souboru a nemusíme tedy vyhledávat informace v dokumentaci. Proměnná této struktury se posílá ve funkcích pro generování konečně-prvkové sítě. Při testování těchto parametru došlo k zjištění, že většina parametru nemá vliv na vygenerovanou konečně-prvkové síť vstupních souboru formátu STL, a proto testování teto knihovny probíhalo spíš na STEP souborech. Časový úsek pro generování konečně-prvkové sítě se u většiny parametrů nijak razantně nemění. Jediná změna přichází při nastavování parametrů maxh a minh. Pokud docílíme nastavením těchto dvou parametrů, že vygenerovaná síť bude jemnější než při základním nastavení, začne se čas generování prodlužovat.

1. maxh - Parametr určující maximální velikost elementu. Funguje na stejném principu jak u knihovny GMSH.

2. minh - Parametr pro určení minimální velikosti elementu. Jak je popsáno výše je podobny parametru z knihovny GMSH. Toto je jediný parametr, který ovlivňoval vstupní soubory formátu STL. Výstup těchto dvou parametru je znázorněn na obrázku 5.6b kde jsou oba parametry nastavené na hodnotu 10. Obrázek 5.6a je v základním nastavení a můžeme jej porovnat se stejnou konečně-prvkovou sítí vygenerovanou knihovnou GMSH 5.5e. Je zde vidět že vygenerovaná síť z knihovny GMSH je jemnější ale doba generování je delší jak je uvedeno v tabulce 5.1.

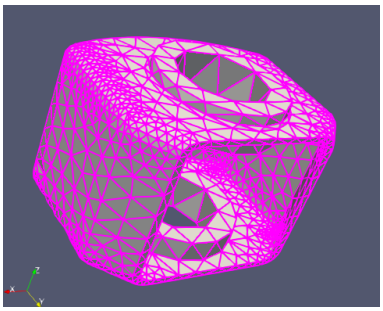
3. granding - Parametr určující rychlost zmenšování konečně-prvkové sítě. Čím víc se hodnota blíží k 0, je vnitřní struktura sítě jemnější, ale povrch zůstává stejný. Porovnat je lze na obrázcích 5.6c, kde je nastavena hodnota parametru na 0, a 5.6d, kde je nastavena hodnota na 1. Tento parametr neprovede žádnou změnu pokud jsou ostatní parametry v základním

nastavení. Je tedy třeba poupravit strukturu vygenerované sítě, ať už k jemnějšímu, tak k hrubšímu výsledku a poté lze tento parametr použít.

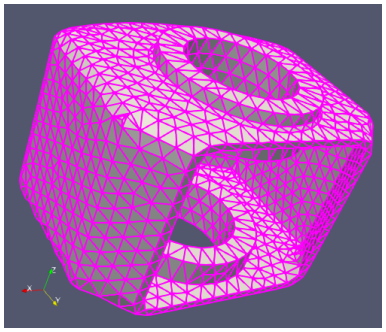
4. elementperedge - Parametr určující počet elementu na hraně objektu. Na obrázku 5.6e je vidět že podél hrany povrchové sítě je 5 elementů. Při nastavování parametru na vyšší hodnotu nastává zlom u hodnoty 9, kdy se počet vygenerovaných elementu sníží na hraně o 1. Tento výsledek můžeme vidět na obrázku 5.6f, kde je hodnota nastavená na 10 a na hraně je pouze 9 elementu. Směrem ke středu plochy se vygenerovaná sít stává hrubší.

5. optsteps3d - Parametr sloužící k optimalizaci vygenerované konečně-prvkové sítě. S rostoucím počtem optimalizací se zlepšuje kvalita vygenerované sítě. Pro Netgen je nastaven v základu na hodnotu 3.

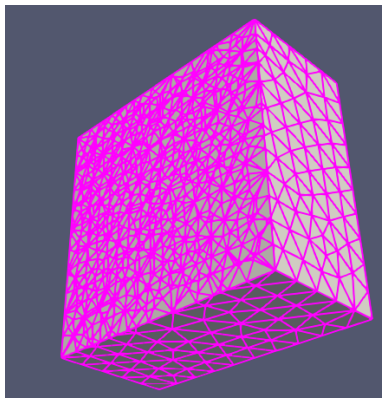
6. refine - Refine je Funkce složí na rozdělení všech elementů a proto není součástí struktury Ng_Meshing_Parameters, ale má vlastní funkci Ng_Uniform_Refinement (Ng_Mesh * mesh), do které se posílá vygenerovaná konečně-prvková sít. Bližší popis tohoto parametru lze najít u knihovny GMSH parametr 11. Stejne jako u předchozí knihovny GMSH je většině případů tady tato funkce nepotřebná, protože výsledná geometrie se lehce ovlivnit výše popsanými parametry.



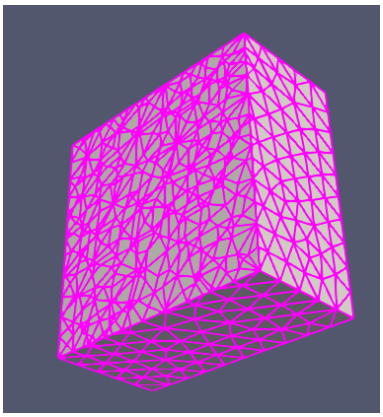
(a) Konečně-prvková sít souboru t13



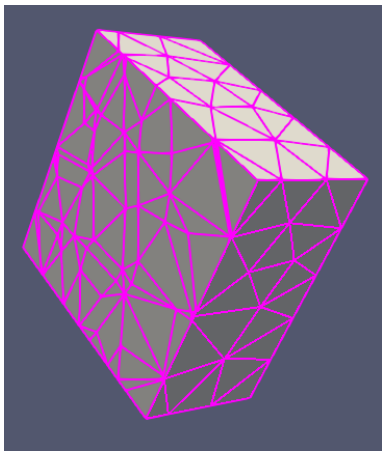
(b) Konečně-prvková sít souboru t13 s parametrem maxh a minh



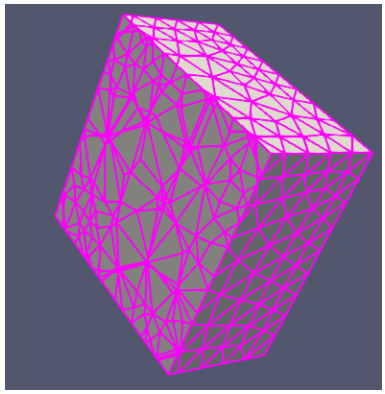
(c) Konečně-prvková sít souboru Cube s parametrem granding na hodnotě 0



(d) Konečně-prvková sít souboru Cube s parametrem granding na hodnotě 1



(e) Konečně-prvková sít souboru Cube s parametrem elementperedge na hodnotě 5

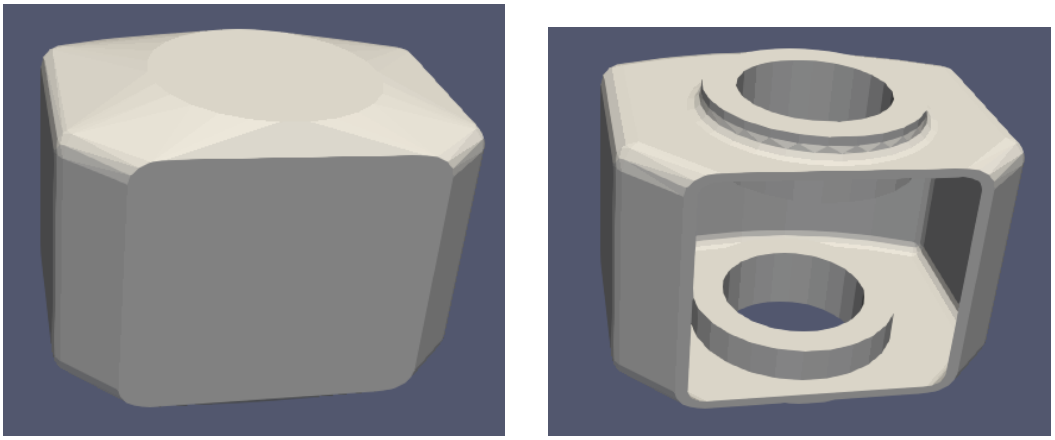


(f) Konečně-prvková sít souboru Cube s parametrem elementperedge na hodnotě 10

Obrázek 5.6: Testované parametry knihovny Netgen

5.3 Tetgen

Jako poslední testovanou knihovnou byla knihovna Tetgen. Tato knihovna také potřebovala pro svou práci přilinkovat hlavičkový soubor *tetgen.h* a knihovnu *libtet.so*. Tyto soubory získáme instalací této knihovny popsanou v dokumentaci v kapitole 4.3. Při práci s touto knihovnou jsme narazili na problém, kdy se nám při práci s vlastním kódem nepodařilo ze vstupní geometrie vygenerovat kvalitní konečně-prvkovou síť a ani při načítání různých parametrů popsaných v dokumentaci jsme nedocílili výsledku, který by byl adekvátní. Knihovnu tedy bylo možně využít jen pomocí příkazové řádky, kde také reagovala na různé nastavení parametrů pro vytvoření jemnější sítě. Na rozdíly těchto dvou případů použití knihovny se můžeme podívat na obrázku 5.7, kde se vlevo nachází síť vygenerovaná naším programem a vpravo síť pomocí příkazové řádky. Tím, že šlo tuto knihovnu využít pouze z příkazové řádky a nepodařilo se nám vygenerovat kvalitní konečně-prvkovou síť, jsme tuto knihovnu nevyužili pro připojení ke konečně-prvkovému řešiči ESPRESO.



(a) Konečně-prvková síť souboru t13.stl pomocí našeho kódu

(b) Konečně-prvková síť souboru t13.stl pomocí příkazové řádky

Obrázek 5.7: Výsledek Tetgen knihovny

Kapitola 6

Napojeni na konečně-prvkový řešič ESPRESO

Knihovna ESPRESO [46] je konečně-prvkový řešič vyvíjený v rámci VšB na IT4Innovations. Tato knihovna nabízí několik řešiců, které cílí na výpočet inženýrských úloh na superpočítačích. Vstupem této knihovny mohly být pouze konečně-prvkově sítě, nad kterými se provedl výpočet dle zadaných fyzikálních vlastností. Hlavním cílem této bakalářské práce bylo rozšíření funkcionalitu o možnost načítat také geometrie zadané v nějakém CAD formátu, jak bude popsáno v teto kapitole.

Knihovna ESPRESO je psaná v C++. Zdrojový kód se skládá z několika modulů. Pro naše účely byly potřeba udělat úpravy v následujících částech: buildování, konfigurace vstupu, načítaní konečně-prvkové sítě a napsat moduly zaobalující volání knihoven třetích stran (GMSH, Netgen). ESPRESO umožňuje v aktuální verzi používat spousty knihoven (Metis, HDF5, ...), proto jsme při dopisování vycházeli z toho jak jsou tyto napojení již napsány.

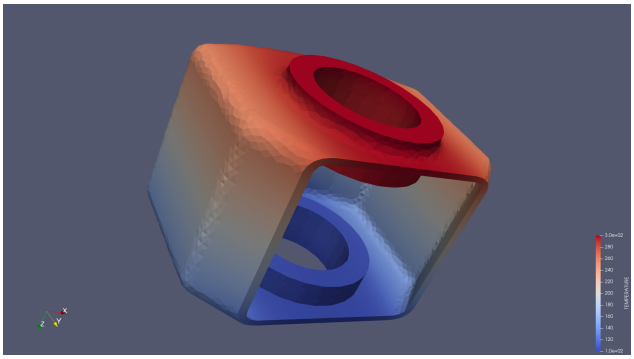
Nejprve bylo potřeba novou knihovnu přidat do buildovacího systému. ESPRESO k tomuto využívá nástroj waf [47]. Řízení je pomoci souboru *wscript* v kořenovém adresáři. Tady stačilo přidat hledání souboru *wscript* ve složce *src/wrappers/gmsh* a *src/wrappers/nglib*, do kterých se přidal kód pro konkrétní knihovny. *wscript* soubory v těchto adresářích již hledaly požadované hlavičky a **.so* soubory. Zde nastal problém s kompatibilitou. Při přilinkování gmsh i netgen knihoven docházelo k pádu aplikace po zavolání netgenu. Proto jsme přidali podmínku, která omezila možnost mít obě knihovny najednou. Potom jsme již nepozorovali žádné problémy.

Běh ESPRESA je řízen textovým konfiguračním souborem (příklady jsou v repozitáři v benchmarks adresáři). Tyto soubory jsou odrazem konfigurace v samotné knihovně. Musely se tedy přidat nové třídy popisující konfiguraci nových knihoven. Toto se provádělo v adresáři *src/config/ecf*, kde se vytvořily nové struktury s parametry popsanými v kapitole (gmsh 5.1.1 a netgen 5.2.1). Po přidaní těchto tříd a jednoduchého popisu jejich parametrů proběhlo již vše automaticky.

Modul pro načítání: opět jsme vycházeli z již hotových modulů. Tady nastal drobný problém s knihovnou netgen, jelikož takový modul již v ESPRESu existoval pro načítání konečně-prvkových databází v tomto formátu. Proto byl netgen přidán jako nglib. Načítací moduly musí korektně vyplnit třídu *MeshData*. Protože všechny parametry této třídy jsou poskytovány generátory, tak se

jen předala reference na tuto třídu do funkcí ve složce *wrappers*, které obsahovaly kód pro voláni jednotlivých generátorů.

Kód pro volání knihoven ve složce *wrappers* vycházel z kódu, který byl vytvořen pro testování. Jen místo do souboru formátu VTK se data kopírovaly do třídy *MeshData*. Konkrétně šlo o parametry id a souřadnice všech bodů a id, velikost, typ a body všech elementů. Bylo tedy nutné změnit typ, který se ukládal do VTK na typ, který interně používá knihovna ESPRESO. Zároveň se vygenerované elementy rozdělily do regionu (pojmenovaných množin), aby se na tyto regiony poté bylo možné odkazovat a zadávat na ně fyzikální vlastnosti dané konečně-prvkové sítě. Na rozdíl od testovacího kódu, kde se parametry měnil ručně, byly hodnoty parametrů nastavené tak, jak je uživatel nastavil v konfiguračním souboru. Vzniklou konečně-prvkovou sít lze tedy editovat bez nutnosti znovu přeložit celou knihovnu. Správné chování bylo ověřeno na souborech, se kterými byly prováděny testy popsané v předchozí kapitole. Na příklad, jak vypadá spočtená úloha, se můžeme podívat na obrázku 6.1.



Obrázek 6.1: Výpočet provedený knihovnou ESPRESO na geometrii t13.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo rozšíření konečně-prvkového řešiče ESPRESO o knihovny, které by převáděly vybrané CAD formáty na konečně-prvkové sítě, se kterými poté tento řešič pracoval dále.

V první řadě bylo potřeba se seznámit s vektorovými formáty a konečně-prvkovými sítěmi, abych pochopil, jak se objekty pomocí vektorové grafiky vytváří a poté jak se vygenerované data ukládají. Dále bylo potřeba zjistit jaké jsou nabízené knihovny pro generování konečně prvkových sítí a vybrat z nich ty, které pro pro naše účely nejvhodnější.

Po této teoretické části následovalo vytvoření aplikace pro testování vybraných knihoven. Při implementaci kódu na testování nastal první problém, kdy nebylo možné mít v jednom testovacím souboru zároveň knihovnu GMSH a Netgen, a proto jsem tyto 2 knihovny musel rozdělit do dvou souborů. Během tohoto testování jsem zkoumal jak vybrané knihovny generují konečně-prvkové sítě a sledoval jejich časové průběhy k zjištění, která z těchto knihoven generuje síť rychleji bez vlivu na kvalitu vygenerované sítě. Během tohoto testování jsem došel k druhému problému kdy jsem nebyl schopen vygenerovat správnou konečně-prvkovou síť pro knihovnu Tetgen, i když při použití příkazové řádky byla knihovna schopna tuto síť vygenerovat a bylo jí také možné upravovat podle vložených parametrů.

Ve výsledku tohoto testování jsem došel k závěru, že knihovna Netgen je časově rychlejší a tento rychlejší průběh generování nemá vliv na kvalitu vygenerované sítě. Knihovna Netgen má také jednodušší použití pro uživatele a není potřeba aby uživatel sám nastavoval parametry na zachycení složitosti vložené geometrie.

V poslední části jsem se zabýval napojení testovací aplikace na konečně-prvkový řešič ESPRESO, kde bylo potřeba změnit způsob ukládání dat, ke kterému se v testovací aplikaci používal soubor formátu VTK, protože řešič ESPRESO měl pro toto vkládání vlastní třídu *MeshData*. Dále jsem musel ještě nastavit aby nebylo možné používat obě knihovny najednou, protože v opačném případě aplikace padala. Poté už bylo třeba jen otestovat správné chování celé aplikace.

Literatura

1. *Computer graphics* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Computer_graphics.
2. *Vector graphics* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Vector_graphics.
3. *Adobe Fireworks* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Adobe_Fireworks.
4. *Sketch (software)* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Sketch_(software).
5. *Inkscape* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Inkscape.
6. *Autodesk Inventor* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Autodesk_Inventor.
7. *AutoCAD* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/AutoCAD.
8. *Vector formats* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Image_file_formats#Vector_formats.
9. *Scalable Vector Graphics* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Scalable_Vector_Graphics.
10. *IGES* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/IGES.
11. *ISO 10303-21* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/ISO_10303-21.
12. *Mesh generation* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Mesh_generation.
13. *Partial differential equation* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Partial_differential_equation.
14. LISEIKIN, V.D. *Grid Generation Methods*. Springer Netherlands, 2009. Scientific Computation. ISBN 9789048129126. Dostupné také z: https://books.google.cz/books?id=fp9TtG2APJ8C.

15. *VTK File Formats* [online] [cit. 2021-04-18]. Dostupné z: https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf.

16. *Ansys EnSight | Simulation Data Visualization Software* [online] [cit. 2021-04-18]. Dostupné z: https://www.ansys.com/products/fluids/ansys-ensight.

17. *Abaqus* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Abaqus.

18. *OpenFOAM* [online] [cit. 2021-04-18]. Dostupné z: https://www.openfoam.com/.

19. *XDMF Model and Format* [online] [cit. 2021-04-18]. Dostupné z: https://www.xdmf.org/index.php/XDMF_Model_and_Format.

20. *Mesh Generation and Grid Generation: Software* [online] [cit. 2021-04-18]. Dostupné z: http://www.robertschneiders.de/meshgeneration/software.html.

21. *Links to free/open source meshing software* [online] [cit. 2021-04-18]. Dostupné z: https://web.archive.org/web/20180726185725/http://graphics.tudelft.nl/~matthijss/oss_meshing_software.html.

22. *ANSA pre-processor - BETA CAE Systems* [online] [cit. 2021-04-18]. Dostupné z: https://www.beta-cae.com/ansa.htm.

23. *Review: Ansys Workbench 15* [online] [cit. 2021-04-18]. Dostupné z: https://develop3d.com/product-design/review-ansys-workbench-15/.

24. *Ansys* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Ansys.

25. *Solid Edge* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Solid_Edge.

26. *Siemens PLM Software* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Siemens_PLM_Software.

27. *Comet Solutions - Simulation Driven Design Apps* [online] [cit. 2021-04-18]. Dostupné z: http://cometsolutions.com/.

28. *MeshLab* [online] [cit. 2021-04-18]. Dostupné z: https://www.meshlab.net/.

29. *CGAL* [online] [cit. 2021-04-18]. Dostupné z: https://www.cgal.org/.

30. *CGAL 5.2.1 - Manual: Package Overview* [online] [cit. 2021-04-18]. Dostupné z: https://doc.cgal.org/latest/Manual/packages.html.

31. *Welcome to the www.salome-platform.org - SALOME Platform* [online] [cit. 2021-04-18]. Dostupné z: https://www.salome-platform.org/.

32. *Open CASCADE* [online] [cit. 2021-04-18]. Dostupné z: https://www.opencascade.com/.

33. *R&D: global expertise | EDF France* [online] [cit. 2021-04-18]. Dostupné z: https://www.edf.fr/en/the-edf-group/inventing-the-future-of-energy/r-d-global-expertise.

34. *CEA - Energies* [online] [cit. 2021-04-18]. Dostupné z: https://www.cea.fr/Pages/domaines-recherche/energies.aspx.

35. *Gmsh* [online] [cit. 2021-04-18]. Dostupné z: https://gmsh.info/.

36. *Netgen/NGSolve* [online] [cit. 2021-04-18]. Dostupné z: https://ngsolve.org/.

37. *TetGen* [online] [cit. 2021-04-18]. Dostupné z: http://wias-berlin.de/software/index.jsp?id=TetGen&lang=1.

38. *Gmsh 4.8.3* [online] [cit. 2021-04-18]. Dostupné z: https://gmsh.info/doc/texinfo/gmsh.html.

39. *Netgen documentation* [online] [cit. 2021-04-18]. Dostupné z: https://github.com/NGSolve/netgen/blob/master/doc/ng4.pdf.

40. *Tetgen documentation* [online] [cit. 2021-04-18]. Dostupné z: http://wias-berlin.de/software/tetgen/1.5/doc/manual/manual001.html.

41. *GetDP* [online] [cit. 2021-04-18]. Dostupné z: https://getdp.info/.

42. *GMSH tutorial* [online] [cit. 2021-04-18]. Dostupné z: https://gitlab.onelab.info/gmsh/gmsh/-/tree/master/tutorial.

43. *Install from sources* [online] [cit. 2021-04-18]. Dostupné z: https://docu.ngsolve.org/latest/install/install_sources.html.

44. *Open CASCADE Technology* [online] [cit. 2021-04-18]. Dostupné z: https://www.opencascade.com/open-cascade-technology/.

45. *Open CASCADE Technology* [online] [cit. 2021-04-18]. Dostupné z: https://dev.opencascade.org/doc/overview/html/.

46. *ESPRESO, Knihovna masivně paralelních řešičů pro inženýrské aplikace* [online] [cit. 2021-04-18]. Dostupné z: http://numbox.it4i.cz/.

47. *The Waf Book* [online] [cit. 2021-04-18]. Dostupné z: https://waf.io/book/.

Příloha A

Popis přiložené přílohy

1. src- Adresář se zdrojovými soubory testovací aplikace
2. test - Adresář s testovacími vstupy
3. wscript - Soubor pro definování knihoven a kompilace souboru pro WAF
4. waf - Kompilační soubor
5. README.txt - Stručný popis k instalaci knihoven pro spuštění testovacího kódu.